Technical Documentary
Report No. ESD-TDR-64-389

MILITRAN
OPERATIONS MANUAL
FOR
IBM 7090-7094

Prepared for the

OFFICE OF NAVAL RESEARCH
NAVY DEPARTMENT
WASHINGTON, D. C.

and the

DIRECTORATE OF COMPUTERS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
L. G. HANSCOM FIELD, BEDFORD, MASS.

U. S. Navy Contract No. Nonr 2936(00)

by

SYSTEMS RESEARCH GROUP, INC.
1501 Franklin Avenue
Mineola, L. I., New York

JUNE 1964

AD0601795

DDC AVAILABILITY NOTICE

Qualified requesters may obtain copies of this report from the Defense Documentation Center (DDC), Cameron Station, Alexandria, Va. 22314. Orders will be expedited if placed through the librarian or other person designated to request documents from DDC.

MILITRAN
OPERATIONS MANUAL
FOR
IBM 7090-7094

Prepared for the

**OFFICE OF NAVAL RESEARCH
NAVY DEPARTMENT
WASHINGTON, D. C.**

and the

**DIRECTORATE OF COMPUTERS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
L. G. HANSCOM FIELD, BEDFORD, MASS.**

U. S. Navy Contract No. Nonr 2936(00)

by

SYSTEMS RESEARCH GROUP, INC.
1501 Franklin Avenue
Mineola, L. I., New York

JUNE 1964

DDC AVAILABILITY NOTICE

FOREWORD

This is one of three technical reports being published simultaneously. The others are the MILITRAN Programming Manual (Technical Documentary Report No. ESD-TDR-64-320) and the MILITRAN Reference Manual (Technical Documentary Report No. ESD-TDR-64-390). The three reports constitute a complete description and instructions for using the MILITRAN language in computer programming of simulation problems.

The MILITRAN 7090-7094 Processor, which is used to compile a problem written in MILITRAN source language into a machine language program, will be available to prospective users. Pending final arrangements, requests for information about the MILITRAN Processor should be sent to the Office of Naval Research (Code 491).

ABSTRACT

MILITRAN is an algorithmic computer language specifically oriented to the problems encountered in simulation programming. In addition to providing over-all flexibility in expressing complex procedures, the language contains features which greatly simplify the maintainence of status lists, handling of numeric and non-numeric data, and sequencing of events in simulated time.

This report describes the features and operating procedures of the 7090-94 MILITRAN Processor.

REVIEW AND APPROVAL

This Technical Documentary Report has been reviewed by the Electronic Systems Division, U. S. Air Force Systems Command, and is approved for general distribution.

J. B. CURTIS
2nd Lt., USAF
PROJECT OFFICER

TABLE OF CONTENTS

Page

INTRODUCTION

This manual describes features of the 7090 MILITRAN Processor and provides instructions for its use. General knowledge of both MILITRAN and the IBM 7090 are assumed on the part of the reader.

The manual is divided into three major sections. The first of these provides an overall view of the processor and its operation. Two subsequent sections cover details of processor operation which apply to the programmer and machine operator respectively.

Full information necessary for compilation, assembly and execution of programs written in the MILITRAN language is contained herein.

THE 7090 MILITRAN PROCESSOR

The 7090 MILITRAN Processor permits the use of MILITRAN Basic Language in programming algorithms for the IBM 7090/7094. The processor translates programs written in MILITRAN into FAP, the 7090/7094 assembly language.

This section of the MILITRAN Operations Manual describes the machine environment required by the processor, the output obtainable from the processor, and the general structure of the processor.

PROCESSOR ENVIRONMENT

The processor has been designed to operate within the framework of procedures commonly in use at scientific installations using the 7090. Hardware and software requirements of the processor are specified in this subsection.

Hardware Requirements

The basic hardware requirement for the processor is an IBM 7090 or 7094 having the following on-line equipment:

1. Two IBM 7607 Data Channels
2. Ten IBM 726 Magnetic Tape Units
3. One IBM 716 Line Printer

Tape units are distributed equally between the channels, and the printer is attached to channel A.

Although the hardware configuration described above is commonly available, the basic design of the processor would permit modifications reducing hardware requirements to an IBM 7090 with seven tape units. An attendant loss in speed and convenience would, of course, accrue from such modification.

Programs compiled by the processor permit the use of an on-line card reader and punch where available.

Software Requirements

The processor operates as a normal "chain-job" under the FORTRAN Monitor System. MILITRAN compilations may be included in ordinary monitor runs without conflict. Use of other monitor sub-systems is not impaired.

Use of tapes by the compiler is in accordance with standard IBM tape assignments. Input/output routines used by compiled programs incorporate a unit assignment table which is easily adjusted to the requirements of a particular installation.

Programs compiled by the processor make use of a library of subroutines called the MILITRAN Object-Time Library. Since one or more of these programs may be required to run a given program, it is anticipated that installations using MILITRAN frequently will wish to add the MILITRAN Object-Time Library to their monitor tapes. Such inclusion requires only that duplication of entry names be eliminated from the combined libraries.

The dependence of the processor upon the monitor system is limited to the use of a few monitor locations and the BSS Loader. Installations using a modified monitor system will in most cases find no need to alter the processor.

Tape Assignments

Tapes used by the processor are tabulated according to function below. The shorter designations in parentheses will be used throughout this manual.

Unit	Function (Short Name)
A1	FORTRAN Monitor System (System Tape)
A2	Symbolic Input Tape (Input Tape)
A3	Listing Output Tape (Listing Tape)
A4	MILITRAN Processor Tape (Chain Tape)
A5	MILITRAN Intermediate Tape 1 (Scratch Tape 1)
B1	MILITRAN Intermediate Tape 2 (Scratch Tape 2)
B2	Dump Program Intermediate Tape (Dump Tape)
B3	MILITRAN Intermediate Tape 3 (Scratch Tape 3)
B4	Punched Output Tape (Punch Tape)
B5	MILITRAN Compiled Output Tape (FAP Tape)

Tapes B2 and B4 are not used by the processor but have been reserved because of their functions within the monitor system. The availability of B2 facilitates dumping in the event of a malfunction, and use of B5 rather than B4 for compiled output enables the programmer to compile and assemble MILITRAN source programs in a single monitor run.

PROCESSOR OUTPUT

The many references to "compilation" and "com-
piled output" in this manual necessitate an early defini-
tion of the processor output. This subsection describes
the items produced during the compilation process. Except
where specifically stated otherwise, the order in which
items are described is the order in which they are produced.

Source Program Listing

The source program being compiled is copied onto
the Listing Tape in an expanded format. This expanded format
separates the statement label, continuation column, the
statement, and source card identification fields for increased
readability. Each line of the source program listing is
numbered in order to provide a reference for possible diag-
nostics.

Errors discovered during the initial processing of
a statement are noted by diagnostic comments immediately below
the offending statement. The statement itself is listed also,
blanks and comments having been removed.

Alphabetic Symbol Table

A list of all names used in the source program is
produced in alphabetical order on the Listing Tape. Each
name is accompanied by the numeric "internal symbol" which
is used to denote that name in the compiled FAP program.

Diagnostics

Source program statements and overall structure are checked at numerous points in the processor. In order to provide the programmer with as much diagnostic information as possible within a single run of the processor, processing is continued to completion regardless of the number of errors found. Erroneous statements are either wholly or partially ignored during compilation.

Although continuation of processing tends to reduce the number of runs required to check out a program, the omission of erroneous statements may result in "false" diagnostics. Each diagnostic comment should therefore be interpreted with this possibility in mind.

If errors have been discovered during generation of the Source Program Listing, a warning flag to that effect is written on the Listing Tape immediately following the Alphabetic Symbol Table.

Errors discovered during intermediate processing are noted on the Listing Tape following the Alphabetic Symbol Table. Where relevant, the line number of the first card of the erroneous statement is given. Similarly, errors discovered during generation of the compiled FAP program are listed immediately following the FAP Program Listing.

System Symbol List

A list of all names used in the source program
which have a pre-defined meaning to the processor is written
on the Listing Tape in order of their numeric "internal
symbols." Certain symbols, such as "RANDOM" and "MINIMUM
INDEX," may appear in this list because of their implicit
use in the source program, even though they do not appear
explicitly.

Numeric Symbol Table

A list of all source program names which do not
have a pre-defined meaning to the processor is written on
the Listing Tape in order of their numeric "internal symbols."
Each symbol is described as to type, mode, storage area, and
dependence upon symbolic dimensions. Dummy variables used as
procedure arguments are so noted. External names are also
given for reference.

External Procedure List

External procedures which are used by the source
program are listed in numeric order. Procedures which appear
in the MILITRAN Object-Time Library are not listed. External
procedure names exceeding six characters in length are trun-
cated to six characters.

Symbolic Dimension List

All symbolic dimensions whose values are to be specified at running time are listed in the order in which they must be presented to the program. Since storage assignment is accomplished at the beginning of main programs only, this list does not appear in compilation of procedures.

Cards specifying symbolic dimension values are prepared directly from this list. Each symbolic dimension value requires one input card.

An initializing value for the system random number generator is constructed from two input values which are treated by the processor as symbolic dimension.

FAP Program Listing

FAP card images comprising the compiled source program are written on the FAP Tape. Each program is preceded by an end-of-file mark, and an end-of-file follows the last program written on the FAP Tape during a given run of the processor.

FAP card images written on the FAP Tape are also copied onto the Listing Tape. The appearance of the statement "SUSPEND FAP LISTING" anywhere in the source program will cause this listing to be deleted.

Printer Comments

When compilation of a source program is initiated by the processor, the legend

"BEGIN MILITRAN COMPILATION"

is printed on-line. At the conclusion of each compilation, the comment

"xx ERRORS IN ABOVE COMPILATION"

appears. The letters "xx" are replaced by the letters "NO" or the number of diagnostic comments issued during processing.

At the conclusion of a processor run, the number of "BEGIN" comments will equal the number of file marks on the FAP tape. The number of "ERROR" comments will equal the number of programs actually compiled.

Pagination

Output on the Listing Tape is separately paginated for each program compiled.

PROCESSOR STRUCTURE

The 7090 MILITRAN Processor operates as a four-link chain job, chain links being stored on the Chain Tape. Processing proceeds in four phases or "passes." Appendix 3 provides a general description of the functions of each pass.

Passes and chain links do not correspond exactly. Passes I and II are in the core simultaneously; Pass III constitutes one link; Pass IV involves two links.

PROGRAMMING THE 7090 IN MILITRAN

This section deals with aspects of processor operation of interest primarily to the programmer. The mechanics of deck preparation for compiling, assembling, and executing MILITRAN programs on the 7090 are described. Programming considerations which apply to the 7090 version of MILITRAN are discussed.

The subsections which follow assume that the reader is familiar with the rudiments of operation under the Fortran Monitor System. Complete information on this system from a programming viewpoint is available in IBM Publication Number C28-6054-2: Reference Manual, 709/7090 FORTRAN Programming System.

PREPARATION OF DECKS FOR COMPILATION

Deck configurations for translation of MILITRAN source programs into FAP are described in this subsection. The basic deck is described pictorially under "Prototype Deck." Details of deck components follow.

Prototype Deck
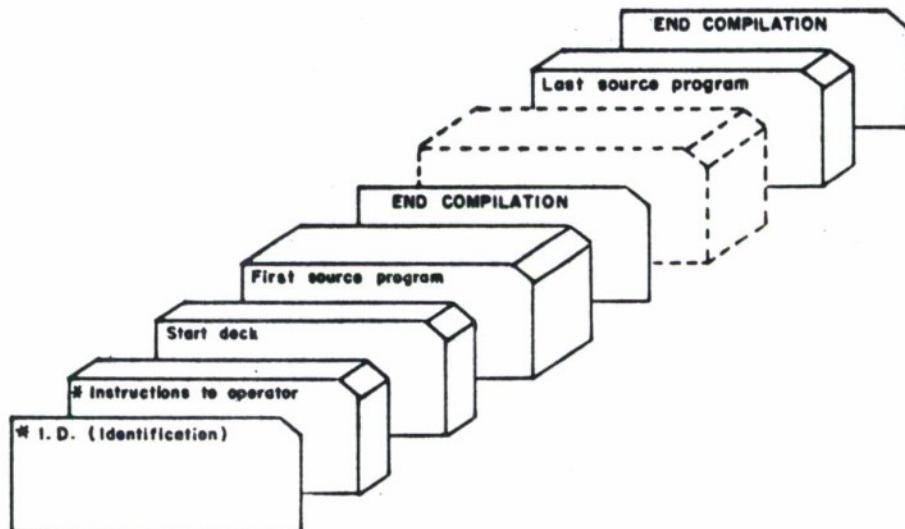
Identification and Instructions

Requirements for the *I.D. card are determined by the accounting routine in use at the machine installation.

Instructions to the machine operator might include a request to mount the MILITRAN Processor Tape and required intermediate tapes; instructions for listing and/or punching of output data off-line; and programmer comments regarding the nature of the run.

The monitor control card "* PAUSE" is usually included to permit required tape handling by the operator.

In general, all identification and instruction
requirements are determined by operating procedures in
effect at each installation.

Start Deck

The "Start Deck" initiates operation of the
processor. A standard Start Deck is provided with the
MILITRAN system and consists of four cards: the monitor
control card "* XEQ"; a two-card binary program which re-
winds the Chain Tape and Listing Tape and initiates
processor operation; and the monitor control card "* DATA."

The binary program which initiates processing
does so through execution of the instruction sequence "CALL
CHAIN (1,t)." Any program which logically concludes with
this sequence may be used in the Start Deck if desired.
The integer "t" should be set to the logical unit designa-
tion of tape unit A4.

Source Programs

Any number of source programs may be processed
during a single processor run, provided that the capacity
of the FAP tape is not exceeded. Each source program must be
terminated by an END COMPILATION statement. Statements which
are erroneously placed between the last END COMPILATION card

and the end of the deck will be listed, but are not compiled as a source program.

A three-character prefix for serialization of FAP output and binary cards is copied exactly from columns 73-75 of the END COMPILATION card for each program compiled. Thus, if source program cards are serialized in card columns 73-80, FAP and binary cards will bear identical codes in columns 73-75.

PREPARATION OF DECKS FOR ASSEMBLY

Two methods are available for assembly of
MILITRAN compiled programs. Card images from the FAP
Tape may be punched off-line and assembled as any other
FAP programs, or assembly may be performed directly
from the FAP Tape. Both methods are discussed in this
subsection.

Assembly from Cards

Programs written on the FAP Tape contain all
information required by the FAP assembler. Only the
monitor control card "* FAP" need be provided for each
program to be assembled. A typical assembly deck is
shown pictorially below.
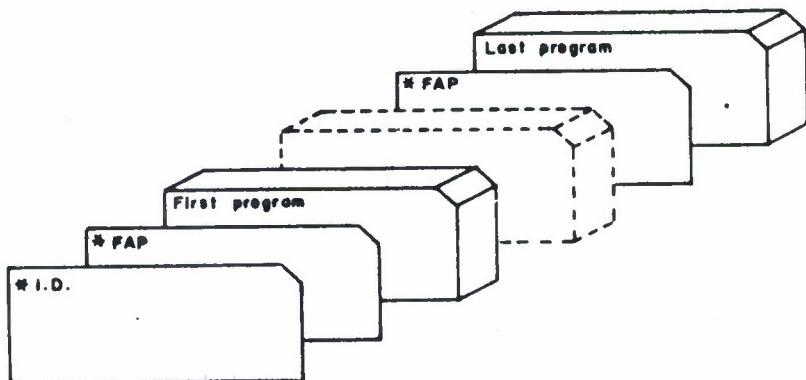
When requesting off-line punching from the FAP Tape, the number of files to be punched should be one greater than the number of programs, since a file mark precedes the first program.

Assembly from Tape

Assembly of programs directly from the FAP Tape is possible through the use of the update features of the FAP assembler. Complete programming information regarding the update facility is available in IBM Publication Number C28-6235: Reference Manual, IBM 709/7090 Programming Systems: FORTRAN Assembly Program [FAP] .

The following sequence of monitor control cards and FAP pseudo-instructions will cause assembly of one program from a properly positioned FAP Tape:

0 1	0 7	0 8	1 5	1 6	7 2	7 3	8 0
*		FAP UPDATE END		t		.	aaa99990

The integer "t" must be the logical unit designation of the FAP Tape, and the prefix "aaa" must correspond to that copied from the END COMPILATION card of the MILITRAN source program.

Proper tape positioning requires that programs be assembled in the order compiled, or that the programmer familiarize himself with the FAP update feature.

Assuming that the FAP Tape is mounted on logical unit 10 and is rewound, the following deck would assemble the first two programs on the tape:
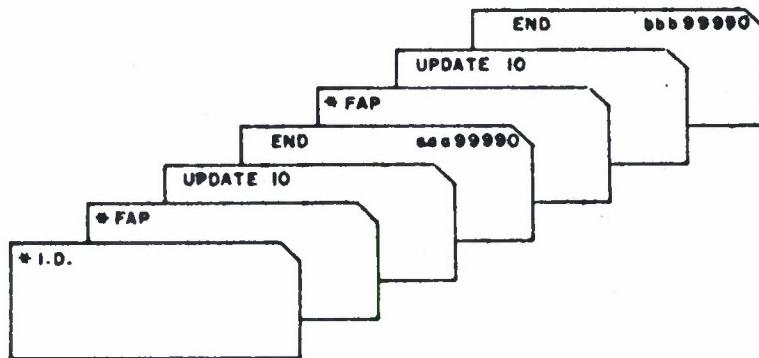
On occasion, it may be desired to assemble only certain programs from the FAP Tape. The following sequence will cause assembly of a particular program from logical unit 10:

0 1	0 7	0 8	1 5	1 6	7 2	7 3	8 0
*		FAP UPDATE SKIPTO END		10		aaa00010 aaa99990	

Again, the prefix "aaa" must be unique and correspond to
the serialization prefix of the program to be assembled.
Proper tape positioning requires that assemblies be per-
formed in order of calculation.

It is recommended that programmers using the
7090 MILITRAN Processor study the use of the update facili-
ty in the FAP manual. Considerable flexibility in handling
compiler output will result.

PREPARATION OF DECKS FOR EXECUTION

Deck configurations for execution of MILITRAN programs are similar to all others used with the FORTRAN Monitor System. A typical configuration is shown pictorially below, and discussion of deck components follows:

Prototype Deck
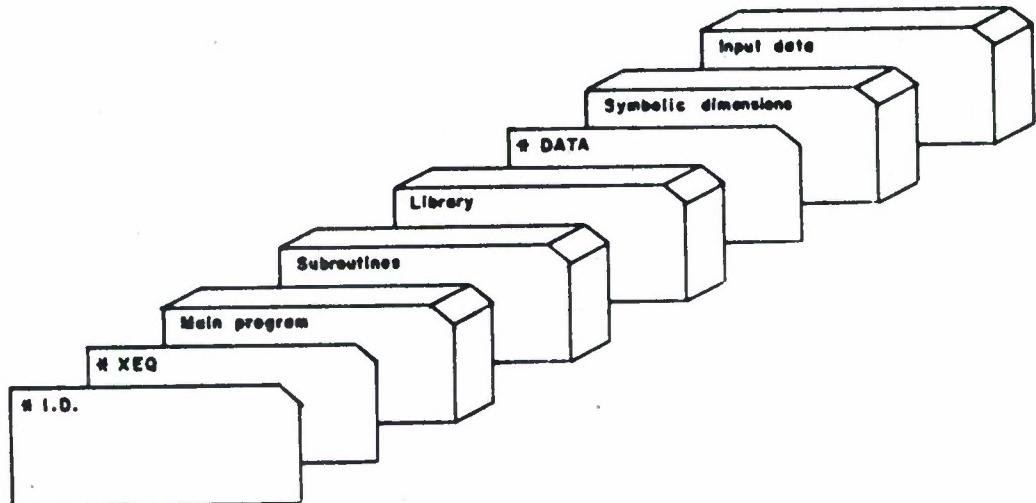
Monitor Control Cards

The three control cards *I.D., *XEQ, and *DATA are required for every run. Additional comment cards may appear after the *I.D. card if desired.

Main Program, Subroutines, and Library

Although the positions of the various binary decks relative to each other are not significant, all binary decks must immediately preceed the *DATA card. One and only one of these decks must be a main program. All others must be subroutines.

A detailed summary of the MILITRAN Object-Time Library will be found in Appendix 1. If this library is included on the installation's System Tape, binary decks for required library subroutines need not be included in the deck configuration. Otherwise, each library subroutine used by the program must be present.

Where computer storage is not at a premium, the entire MILITRAN Object-Time Library may be included in the deck. This eliminates the need for determining the specific subroutines required by the program. Inclusion of the entire library uses approximately 4200 core locations.

Symbolic Dimensions

Cards bearing symbolic dimension values must appear in exactly the same order as the Symbolic Dimension List produced during compilation. Each card must contain a decimal integer, the integer being right-justified in card columns 1-5. Card columns 6-80 are ignored by the program.

Input Data

Input data format is determined entirely by the MILITRAN programmer. The first card of input data must immediately follow the last symbolic dimension card, or the *DATA card if no symbolic dimensions are required.

EXAMPLES OF TYPICAL MILITRAN DECKS

The examples of deck configurations given in this subsection are intended as guides to the programmer in arranging MILITRAN programs for compilation, assembly, and execution. In all cases it is assumed that desired processing is to be accomplished in a single monitor run.

Compile One Program

The example below illustrates a deck to compile a single program. FAP output will be serialized with the prefix "SRG".
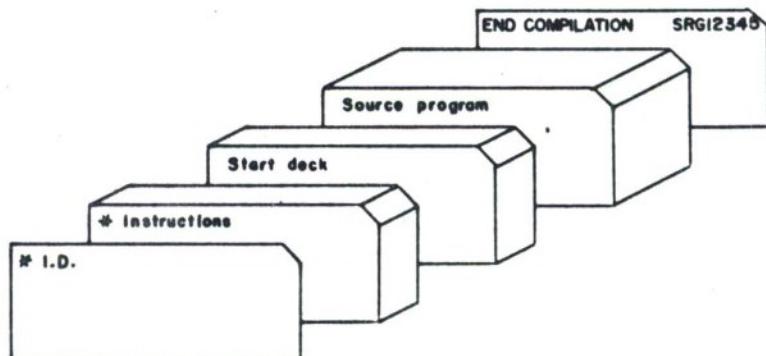
Compile and Assemble Two Programs

The example below illustrates a deck configuration to compile and assemble two programs. END COMPILATION cards are not serialized, and FAP cards will therefore have blanks in card columns 73-75.
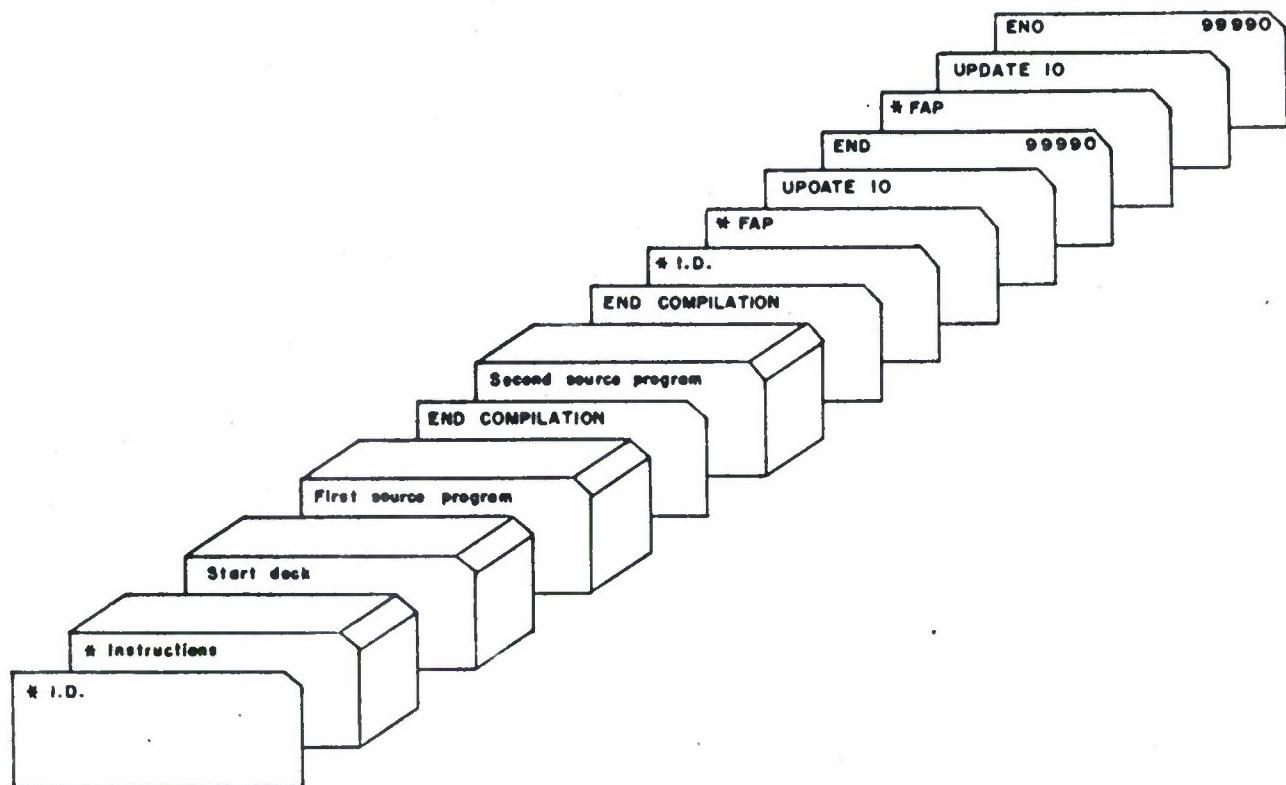
Compile Three Programs, Assemble Two

The example below illustrates a deck configuration to compile three programs and assemble the first and last of these. Card serialization is obviously necessary in this instance.
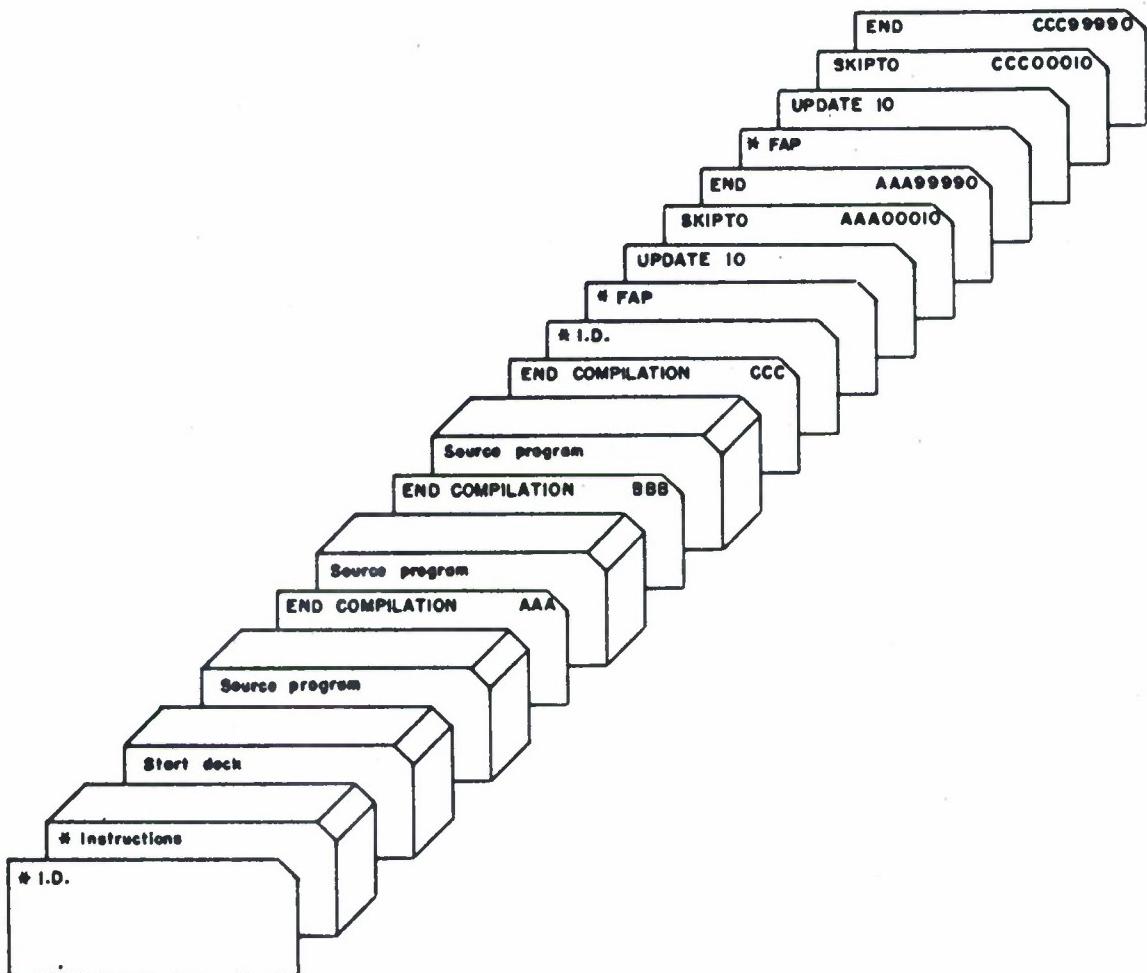
Compile, Assemble, and Execute

      The example below illustrates a deck configuration to compile, assemble, and execute a main program and two subroutines. The END COMPILATION cards are not serialized; the entire library has been included; and no symbolic dimensions have been used.

Input data

* DATA

Library (binary)

END   99990

UPDATE 10

* FAP

END   99990

UPDATE 10

* FAP

END   99990

UPDATE 10

* FAP

* XEQ

* I.D.

END COMPILATION

Subroutine 2

END COMPILATION

Subroutine 1

END COMPILATION

Main program

Start deck

* Instructions

* I.D.
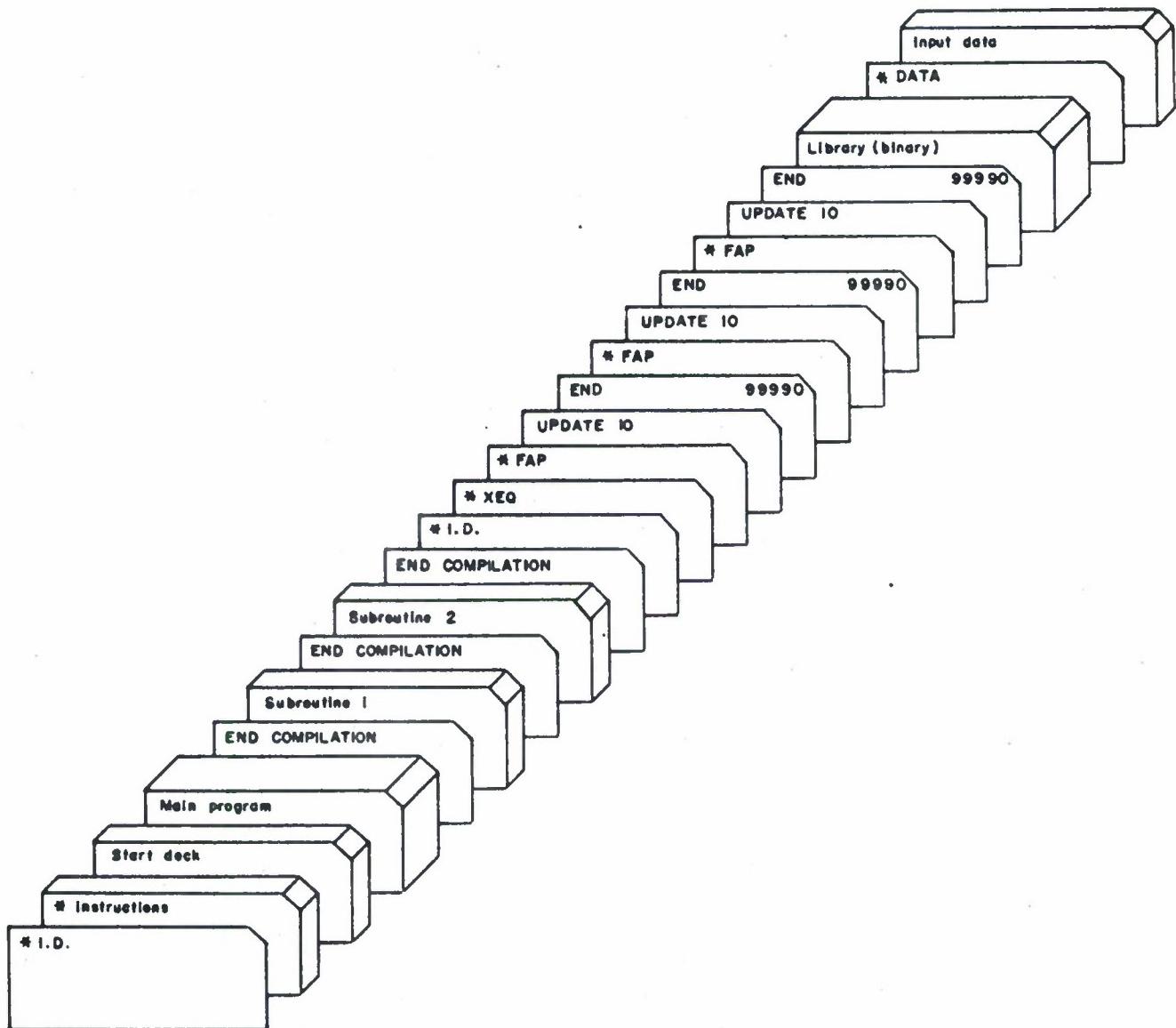
CHAIN JOBS

The CHAIN feature of the standard monitor system may be used to permit execution of programs whose storage requirements exceed the size of the computer. Decks for execution of CHAIN jobs are set-up in MILITRAN in the same manner as in FORTRAN. The sequencing statement "CALL CHAIN (R,T)" used in FORTRAN is replaced in MILITRAN by "EXECUTE CHAIN (R,T)".

Certain characteristics of MILITRAN programs dictate the following considerations with respect to chaining:

1. Symbolic dimensions, if used, should appear in COMMON.

2. Only one main program should contain symbolic dimension references or the system function "RANDOM". This main program should be entered only once.

3. Use of arithmetic expressions as dimensions should be avoided.

As program structure is readily arranged to comply with the above rules, no limitation upon chaining procedures is implied.

USE OF NON-MILITRAN CODING

Under some circumstances, the programmer may wish to incorporate non-MILITRAN coding sections in a MILITRAN program, or use MILITRAN to generate portions of non-MILITRAN programs. Information contained in this subsection will facilitate "hybrid" programming where desired.

Insertion of FAP Coding in MILITRAN Programs

The output of the 7090 MILITRAN Processor is a FAP program embodying the algorithm originally expressed in MILITRAN. Alterations to this program may be made in FAP provided that the coding sequences of the original program are understood.

Variable names in the compiled program have the form "(n)," where "n" is the internal symbol assigned by the processor. Correspondence between internal and external names is found in both the Alphabetic and Numeric Symbol Tables produced during compilation. Temporary storage locations are denoted by ".Tx. - n", where $n \geq 1$ and $1 \leq x \leq 4$.

Access to data in arrays, vectors, and lists
is accomplished by means of indirect addressing through
"declaration tables." The structure of these tables is
summarized in Appendix 2. Relative positions of data are
carried in Index Register 2. Index registers are saved
where possible within a given source statement, but are
not generally carried from one source statement to another.

Subroutine Calling Sequences

The source statement

EXECUTE SUBR(A,B,C,D)

in a MILITRAN program will produce a FAP sequence of the
form

```
TSX     $SUBR,4
BRN     (a)
BRN     (b)
BRN     (c)
BRN     (d)
```

where (a,b,c,d) are the internal symbol numbers corre-
sponding to (A,B,C,D).

FAP coded subroutines compatible with the above calling sequence may be executed by MILITRAN programs. Where all arguments are single arithmetic variables, FORTRAN II and MILITRAN subroutines may also be used together. It should be remembered, however, that integer values stored by FORTRAN II programs will appear to be 262,144 times their actual value when interpreted by MILITRAN programs. This factor of 2^{18} is due to the fact that FORTRAN II does not utilize the full word length in manipulating integer quantities.

MACHINE DEPENDENCE

The MILITRAN Basic Language is generally independent of hardware characteristics. Some features embodied in the processor are, however, peculiar to the 7090/7094, and some programming techniques may limit machine independence.

Machine Oriented Features

The statement "SUSPEND FAP LISTING" and serialization prefix specification on the "END COMPILATION" card are features of the 7090 MILITRAN Processor. They should not be considered as elements of the MILITRAN Basic Language.

Input/output statements such as "UNLOAD" are meaningful only when tape drives permit such operations to be executed by the computer program.

Auxiliary listings produced during compilation are features of the processor alone.

Programming Techniques

Use of non-MILITRAN coding will obviously limit the case with which a program can be implemented on another

computer. This limitation is therefore incurred with the specific knowledge of the programmer.

A more subtle form of machine dependence arises as the programmer gains experience with the processor and the language. It is only natural that certain cause-and-effect relationships will be noted between MILITRAN Source Programs and the FAP programs produced from them. When the programmer takes advantage of his familiarity with the processor in coding MILITRAN programs, loss of machine independence may result.

All possible techniques of source language coding cannot be anticipated in the design of a processor, and even grotesque distortions of a language may at some time be employed to advantage. Considerable effort has been devoted to keeping the language and the processor free of unnecessary restrictions. Where machine independence is of secondary importance, the programmer is free to break rules.

OPERATING THE 7090 MILITRAN PROCESSOR

This section deals with aspects of processor operation of interest primarily to the machine operator. Tape requirements, overall processing sequence, and error procedures are discussed.

GENERAL OPERATING PROCEDURES

In general, the processor operates as any other monitor job. Specific operating characteristics are discussed in this subsection.

Tape Set-up

The processor requires mounting of the MILITRAN Processor Tape on A4, and scratch tapes on A5 and B5. The Processor Tape should be file protected.

All required rewinding of tapes is performed by the processor. In addition, A4 will unload at the end of a processor run. Since other monitor functions may use A4, this feature provides additional protection for the processor and convenience in tape handling for the machine operator.

If the end-of-tape mark is passed while writing A3, and end-of-file is written and A3 unloads. The processor halts after printing a request for a new A3. Processing resumes when START is pressed.

Keys, Sense Switches, Sense Lights

No sense switches are read by the processor at any time. Sense lights are used only to indicate the current processing phase. Sense light 1 is on during Pass I; sense light 2 during Pass II; etc.

All keys must be clear during normal operation of the processor. Keys are used to specify checkout and error procedures only.

Printer Comments

Printer comments are used to indicate the progress of processing, to request a new A3, and to indicate errors in reading the MILITRAN Processor Tape. The comments are self-explanatory.

Processing Sequence

The overall sequence of processor operation is as follows:

1. An initialization program
   (Start Deck) is loaded into core
   and executed.

2. Passes I & II are loaded from A4
   and processing begins.

3 Intermediate tapes B1, B3, and
   A5 are rewound. An end-of-file
   is written on B5. The comment
   "BEGIN MILITRAN COMPILATION" is
   printed on-line.

4. Source statements are read from
   A2 until an "END COMPILATION"
   statement is encountered. If an
   end-of-file appears before "END
   COMPILATION", the processor re-
   winds B5, unloads A4, and returns
   control to the monitor.

5. If no end-of-file is encountered,
   processing proceeds through Pass
   II and Pass III is loaded from A4.

6. When Pass III is complete, Pass IV
   is loaded and A4 rewinds.

7. Pass IV ends by printing a statement on-line which contains the number of errors discovered during processing.

8. If no errors have been discovered by Pass IV, control returns to step 2. Otherwise, the Pass IV Diagnostic Processor is loaded from A4 and executed.

9. A4 is rewound, and control returns to step 2.

ERROR PROCEDURES

This subsection describes procedures to be followed in the event of malfunctions.

Data Channel Traps

The processor uses buffered input/output operations during all passes. As a result, attempts to use normal monitor functions in dumping or jettisoning a run may be frustrated by enabled traps. If non-MILITRAN error procedures are attempted, press RESET in order to clear existing trap signals before starting the computer.

End-of-Tape Marks

The processor will not permit writing beyond the end-of-tape mark on any tape. If the end of any tape except A3 is encountered, the computer stops at octal location 00025. Processing cannot be continued. Press RESET and jettison the run.

Redundancy

The processor will attempt to read or write a record ten times before declaring a tape error. In the event of a tape error, the computer stops at octal location 00024. Processing cannot be continued. Press RESET and jettison the run.

Malfunction

Machine errors or unusual source program errors may cause the processor to stop or loop. When such a condition occurs, the following standard procedure should be followed:

1. Switch computer to MANUAL.

2. Put keys S,1,2,31 and 35 DOWN.

3. Copy the instruction counter into keys 3 thru 17.

4. Press ENTER INSTRUCTION.

5. Switch computer to AUTOMATIC.

6. Press START.

7. When writing has begun on A3, clear the keys.

The above procedure will cause the contents of the computer memory to be dumped onto A3 and control returned to Pass I.

APPENDIX 1

MILITRAN OBJECT-TIME LIBRARY

The MILITRAN Object-Time Library consists of
binary subroutine decks which perform standard process-
ing functions during execution of MILITRAN programs.
The library is divided into three sections as follows:

ML1: Arithmetic Functions

ML2: Input/Output

ML3: System Procedures

Library decks are identified by four-character
codes in columns 73-76 of the binary cards. The two tables
included in this appendix summarize the characteristics of
each deck with respect to MILITRAN source codes.

The LIBRARY CONTENTS table lists for each library
deck its identification code, core storage requirements
(decimal), entry-point names, other library decks used, and
source program characteristics which require its inclusion
at running time.

The LIBRARY USAGE table lists various source pro-
gram characteristics and the library routines required at
running time in order to implement those characteristics.

LIBRARY CONTENTS

Deck Identification	Storage Required	Entry Names	Other Decks Required	Source Program Characteristics
ML1A	30	)EXP1	none	A.P.B, where A and B are both INTEGER.
ML1B	36	)EXP2	none	A.P.B, where A is REAL, B is INTEGER.
ML1C	112	)EXP3	none	A.P.B, where A is REAL, or INTEGER, B is REAL.
ML1D	47	)EXP	none	EXP(A)
ML1E	47	)LOG	none	LOG(A)
ML1F	69	)ATAN	none	ATAN(A,B)
ML1G	121	)SIN)COS)TAN	none	SIN{A} COS{A} TAN{A}
ML1H	45	)SQT	none	SQT(A)
ML1J	43	)RAN	none	RANDOM, or RANDOM INDEX which does not place a GST or LST condition on the entry index.
ML2A	45	)RHN)RHD)RHNM)RHDM	ML2B ML2P ML2T ML2V	READ, READWRITE, RANDOM, or use of symbolic dimensions.
ML2B	126	(CSH)	ML2P ML2S ML2T ML2V	Used by other library routines.
ML2C	91	)WHN)WHD)WHNM)WHDM	ML2D ML2E ML2P ML2Q ML2U ML2V	WRITE, READWRITE, RANDOM, or use of symbolic dimensions.

Deck Identification	Storage Required	Entry Names	Other Decks Required	Source Program Characteristics
ML2D	96	(SCH)	ML2P ML2V	Used by other library routines.
ML2E	183	(SPH)	ML2P ML2V	Used by other library routines.
ML2F	55	)WRB)WLR	ML2N ML2P ML2Q ML2U	BINARY WRITE
ML2G	59	))RDB))RLR	ML2N ML2P ML2T	BINARY READ
ML2H	8	)EFT	ML2P	END FILE
ML2J	8	)RWT	ML2P	REWIND
ML2K	8	)UNL	ML2P	UNLOAD
ML2L	29	)RBS	ML2P	BACKSPACE
ML2M	8	)FBS	ML2P	BACKSPACE FILE
ML2N	366	(IOB) (EXB) (IOBX) (VLSW) (ZEDA) (BUF)	ML2P	Used by other library routines.

Deck Identification	Storage Required	Entry Names	Other Decks Required	Source Program Characteristics
ML2P	91	(IOS RDS WRS BSR WEF REW BSF RUN ETT RCH TEF TCO TRC)	ML2Q ML2R ML2S	Used by other library routines.
ML2Q	1	(TES)	none	Used by other library routines.
ML2R	24	(IOU)	none	Used by other library routines.
ML2S	9	(EXE)	none	Used by other library routines.
ML2T	42	(RER RDC EOFR EORR)	ML2P ML2S	Used by other library routines.

Deck Identification	Storage Required	Entry Names	Other Decks Required	Source Program Characteristics
ML2U	57	(WER) (WTC)	ML2P ML2Q ML2S	Used by other library routines.
ML2V	1419	(IOH) FIL RTN (BLNK) (HOLL)	ML2P ML2S	Used by other library routines.
ML3A	105	)RE1)RE2)RE3)RE4	none	PLACE, PLACE ENTRY, REPLACE, REPLACE ENTRY, REMOVE, REMOVE ENTRY.
ML3B	16	)STOP	none	STOP
ML3C	62	)NEV)ECVL	ML3B	CONTINGENT EVENT
ML3D	34	)LP1	none	MINIMUM INDEX with no GST or LST. MINIMUM INDEX with LST condition on entry index. RANDOM INDEX with LST condition on entry index.
ML3E	19	.)LP2	none	MINIMUM INDEX with GST condition on entry index. RANDOM INDEX with GST condition on entry index. REMOVE with no GST or LST. REPLACE with no GST or LST.

Deck Identification	Storage Required	Entry Names	Other Decks Required	Source Program Characteristics
ML3F	45	}}LP3 LP4	none	MINIMUM INDEX with GST or LST condition on component.
ML3G	65	)LP5	ML1J	RANDOM INDEX with no GST or LST
ML3H	81	}}LP6 LP7	ML1J	RANDOM INDEX with GST or LST condition on component.
ML3J	76	)LP8)LP9	none	REMOVE with GST or LST condition on component. REPLACE with GST or LST condition on component.
ML3K	177	}}DISC OBJ1 OBJ2	none	Use of ARRAYS, VECTORS, LISTS, EVENTS, OBJECTS, or CLASSES.
ML3L	192	}}LV1 LV2 LV3	ML2S	Use of symbolic dimensions.
ML3M	12	)CHAIN	none	Use of chaining feature.

45

Features below, w										ML3									
library	M	N	P	Q	R	S	T	U	V	A	B	C	D	E	F	G	H	J	K	L
Arrays, vectors, lists, events																			X	
Symbolic dimensions			X	X	X	X	X	X	X											X
CONTINGENT EVENT, NEXT EVENT,											X	X								
STOP										X										
A. P. B. where A and B are b																				
where A is REAL and																				
where B is REAL																				
EXP(A)																				
LOG(A)																				
ATAN(A,B)																				
SIN(A), COS(A), or TAN(A)																				
SQRT(A)																				
RANDOM			X	X	X	X	X	X	X											
READ			X	X	X	X	X		X											
WRITE			X	X	X	X		X	X											
READWRITE			X	X	X	X	X	X	X											
BINARY READ		X	X	X	X	X	X													
BINARY WRITE		X	X	X	X	X		X												
END FILE			X	X	X	X														
REWIND			X	X	X	X														
UNLOAD			X	X	X	X														
BACKSPACE			X	X	X	X														
BACKSPACE FILE	X		X	X	X	X														
PLACE										X										
PLACE ENTRY										X										
REMOVE ENTRY										X										
REPLACE ENTRY										X										

| Features below, when require library de-		M	N	P	Q	R	S	T	U	V	A	B	C	D	E	F	G	H	J	K	L
												ML3									
MINIMUM INDEX	with no GST or LST													X							
	with GST referring															X					
	with LST referring															X					
	with GST referring														X						
	with LST referring			X	X	X	X	X	X	X				X			X				
RANDOM INDEX	with no GST or LST			X	X	X	X	X	X	X								X			
	with GST referring			X	X	X	X	X	X	X					X			X			
	with LST referrin														X						
	with GST referring										X				X						
	with LST referring										X								X		
REMOVE	with no GST or LST										X								X		
	with GST referring										X				X						
	with LST referring										X			X							
	with GST referring										X				X						
	with LST referring										X								X		
REPLACE	with no GST or LST										X								X		
	with GST referring										X				X						
	with LST referring										X			X							
	with GST referring																				
	with LST referring																				

APPENDIX 2

DECLARATION TABLES

All objects, classes, arrays, vectors, and lists are fully described in the object program by means of reference tables. These tables are described in detail on succeeding pages.

Objects

	P 2 3	17 18	20 21	35
b-m	Last word of object name			

. . .

b-1	First word of object name			
b	4	d	0	a
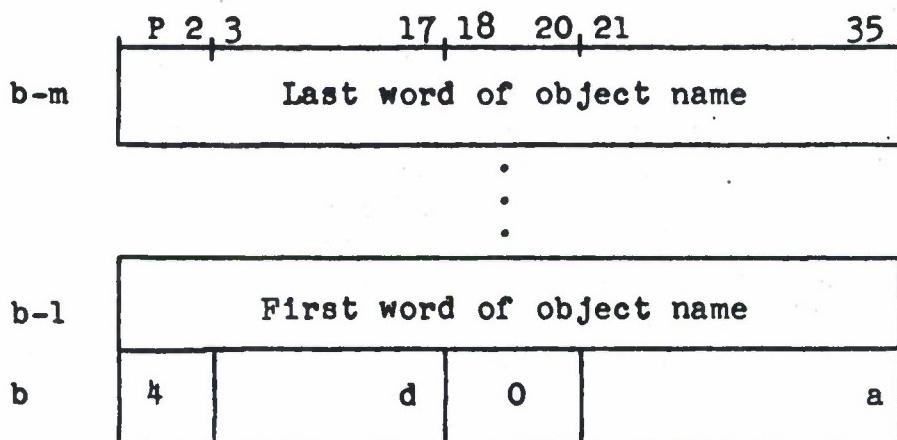
Base address b is the location corresponding to the internal symbol of the object name. The name of the object is stored in BCD code, six characters to the word. The last word of the name is filled with blanks on the right if less than six characters long. Decrement d is the number of characters in the object name, excluding blanks.

If the object was declared with a symbolic dimension, address a is the location of the symbolic dimension value. The actual value replaces "a" after dimension values are read.

If the object was declared with a numeric dimension, a is the value of that dimension.

Classes

	P 2,3	17,18	20,21	35
b-n	P_n	d_n	0	a_n
b-n+1	P_{n-1}	d_{n-1}	0	a_{n-1}

.
.
.

b-2	P_2	d_2	0	a_2
b-1	P_1	d_1	0	a_1
b	0	n	0	c

Base address b corresponds to the internal symbol of the class name. The number of words in the table, excluding the base address, is n. The cardinality, c, is initially zero if any member of the class involves symbolic dimensions. After symbolic dimensions have been read, c is set to the proper cardinality.

Table words in location b-1 through b-n specify the members of the class. Decrements d_1 through d_n are codes designating objects which belong to the class, each object being designated by the base address of its declaration table in core. Addresses a_1 through a_n are the

respective contributions of objects d_1 through d_n to the total cardinality of the class. Prefix p_1 is zero if object d_1 is included collectively (no EACH*) in the class; p_1 is four if object d_1 is included individually (EACH*).

Arrays

	P 2,3	17	18 20,21	35
b-n	P_n	d_n	0	a_n
b-n-1	P_{n-1}	d_{n-1}	0	a_{n-1}
		·		
		·		
		·		
b-2	P_2	d_2	0	a_2
b-1	P_1	d_1	0	a_1
b	0	n	t	a_0

Base address b corresponds to the internal symbol of the array name. Entries in locations b-1 through b-n specify the n dimensions of the array. Address a_0 is the BES address of the array itself. The location specified by a_0 is also reserved, and is called the "zero position."

Arrays are stored in decreasing storage locations from a_0, the first subscript varying most rapidly. Access to arrays is accomplished indirectly by means of the tag t which is either 2 or is set to 2 by the storage allocation processing.

If the ith dimension is numeric, p_i is zero, d_i is zero, and a_i is the dimension value. If the ith dimension is numeric, p_i is 4, d_i is zero, and a_i is the location of the dimension value. The dimension value itself replaces a_i during storage allocation.

If the ith dimension is an object or class name which does not involve symbolic dimensions, p_i is zero, d_i is the base address of the object or class in storage, and a_i is the cardinality of the object or class. If the object or class is dependent upon symbolic dimensions, p_i is 4 and a_i is zero. The cardinality of the class or object as determined by its symbolic dimension values replaces a_i during storage allocation.

Vectors

	P 2 3	17 18 20 21	35
$b-m(n+1)$		$m\underline{\text{th}}$ component	
$b-(m-2)(n+1)-1$		$(m-1)\underline{\text{th}}$ component	

.

.

.

$b-n-2$		Second component		
$b-1$		First component		
b	4	m	t	a_0
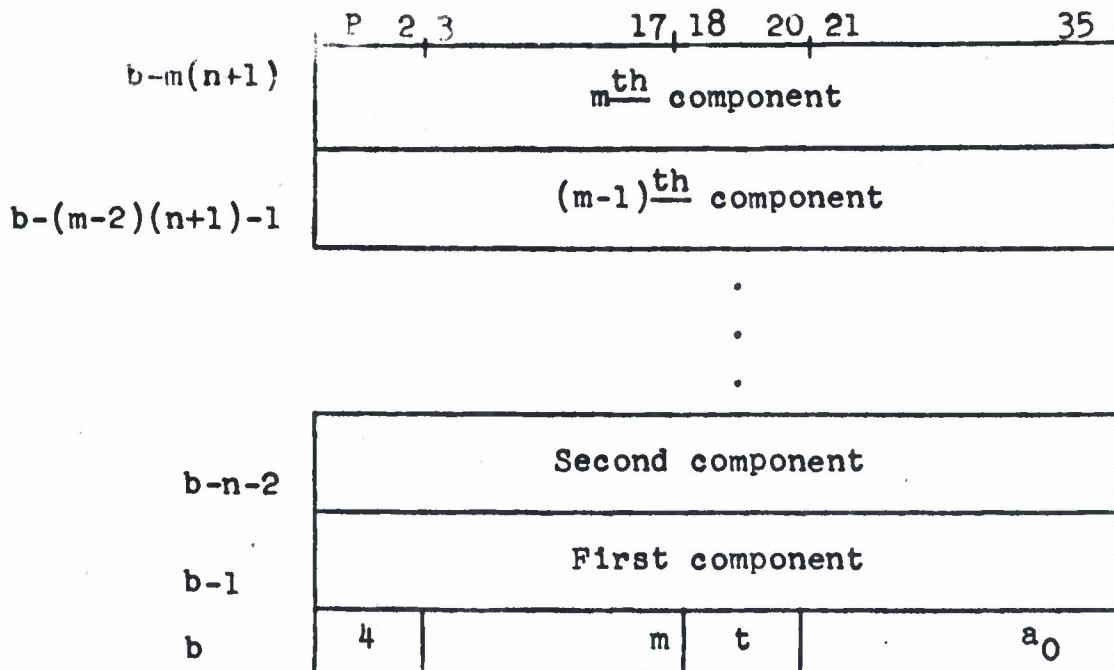
Each of the m components has n dimensions and its declaration table is identical to that for an array. Only one "zero position" is reserved for the entire vector, and that position is at a_0. Tag t is either 2 or is set to 2 during storage allocation.

Vectors are stored in decreasing storage locations from a_0. The first component has its "zero position" at a_0; the second component shares its "zero position" with the last position of the first component; etc.

Base address b corresponds to the internal symbol of the vector name. The base addresses of array tables for components correspond to the internal symbol of the component names.

Lists and Permanent Events with Lists

	P 2,3		17,18	20,21	35
b-3m			Vector table		
b					
b+1	0	0		0	a_{-1}
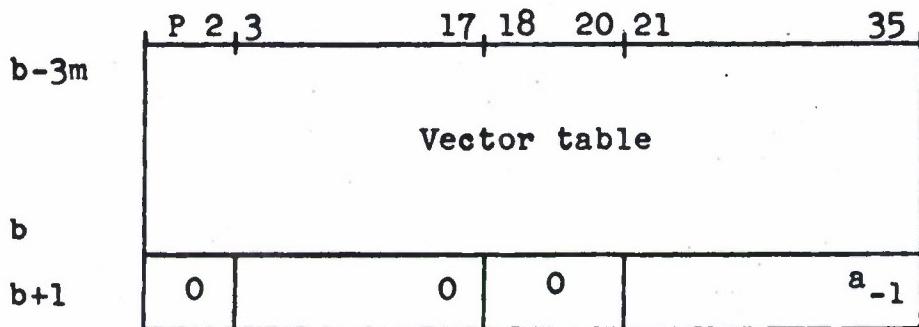
The "vector table" is identical to that for a vector having m components of one dimension each. The address a_{-1} at location b + 1 is the current length of the list. This length is originally zero.

Base address b corresponds to the internal symbol of the list name.

Contingent Events with Lists

	P 2 3		17 18	20 21	35
b-3m					
			Vector table		
b					
b+1	4		0	0	a_{-1}
b+2	0		0	0	a_{-2}
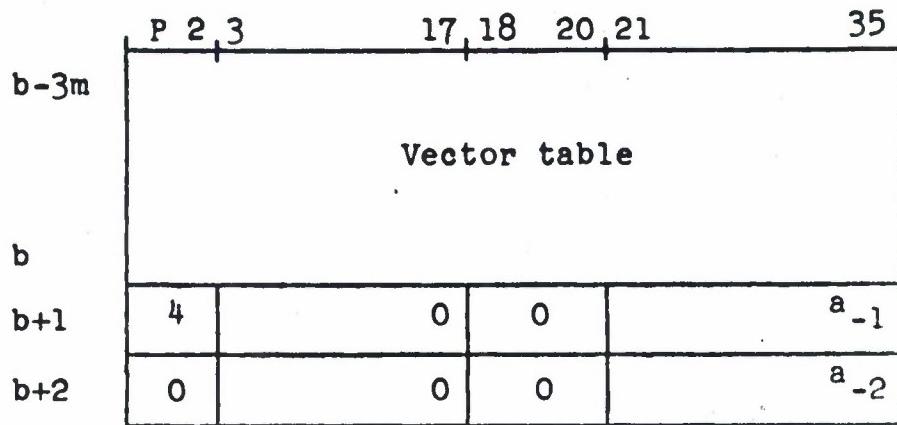
Location $b + 2$ specifies the address (a_{-2}) of the first instruction in the event processing sequence. Other locations in the table are as described under Lists.

APPENDIX 3

FUNCTIONS OF PROCESSOR PASSES

Functions of each pass are described below. Passes are executed sequentially for each program compiled. Initiation and termination of the operation of the processor occurs in Pass I.

Pass I

All reading of the Input Tape is performed by Pass I. Statements are numbered and copied onto the Listing Tape as they are read. Buffered input/output operations permit these functions to occur concurrently with internal processing.

Each statement is subjected to a general scan which eliminates blanks and comments. FORMAT and END COMPILATION statements are identified immediately and processed separately. Diagnostics issued during preliminary scanning are followed by a listing of the scanned statement up to the point at which the error is discovered.

After preliminary scanning, statement types are identified and statements are reduced to an internal code for

processing. All names are assigned unique numeric identifiers which will become the basis for internal storage of their declared characteristics. Mnemonic delimiters such as UNTIL, FOR, and BY are identified.

Final processing in Pass I is determined by the type of statement. Symbolic dimensions are identified and declared. Statement labels are declared. FORMAT statements are compressed and translated into final form.

Declarative statements are written on Scratch Tape 2; executable statements are written on Scratch Tape 3; certain "hybrid" statements are written either wholly or partially on both tapes. Names and their internal numeric counterparts are written on Scratch Tape 1.

When the END COMPILATION statement has been processed, the Alphabetic Symbol Table is written on the Listing Tape and Pass I processing ends. Ends-of-file are written on all scratch tapes and Scratch Tape 2 is rewound.

Pass II

Declarative statements are read from Scratch Tape 2 and processed. Explicit declarative information is recorded in the symbol table for later reference. Names

declared as vectors, lists, or events are assigned the NORMAL MODE if specific mode declarations are not encountered. FORMAT statements are sorted onto Scratch Tape 3; expressions involving symbolic dimensions onto Scratch Tape 1. Diagnostics are issued as required.

When all declarations have been processed, the symbol table is scanned and adjusted to provide full information regarding each explicitly declared symbol. Diagnostics are issued without source line references.

Ends-of-file are written on Scratch Tapes 1 and 3; Scratch Tapes 2 and 3 are rewound; Scratch Tape 1 is backspaced to the beginning of the second file.

Pass III

All executable statements on Scratch Tapes 1 and 3 are processed by Pass III. Each symbol whose mode has not explicitly been defined is assigned the NORMAL MODE. Functions are identified. The internal code in which the statement is expressed is modified to facilitate further processing.

Statements are examined by type, and complex statements are rewritten in terms of more basic operations. Diagnostics are issued as required. Rewritten statements

are checked for validity of internal code symbols and copied onto Scratch Tape 2.

The executable program written on Scratch Tape 2 retains the basic syntax of the MILITRAN source program. However, certain statement types not found in the external language have been introduced, and many external statement types have disappeared. Some temporary storage allocation has been determined; some auxiliary labels have been introduced. Constants retain the apparent mode specified by the presence or absence of a decimal point in their external form.

When all executable statements have been processed, the System Symbol List is generated. External names are retrieved from the first file of Scratch Tape 1 and the Numeric Symbol Table is written. Names of procedures, objects, and symbolic dimensions are retained for later reference; all others are discarded. The External Procedure List and Symbolic Dimension List are written. Scratch Tapes 1 and 2 are rewound.

Pass IV

Pass IV processing consists entirely in generating a FAP program on the FAP tape. Programs generated have the following canonical form:

1. A "LBL" pseudo-instruction causing binary cards to be serialized with columns 73-75 of the END COMPILATION card.

2. A "COUNT" pseudo-instruction whose address field is seven times the number of cards in the source program.

3a. A page title card and an initial transfer instruction. (Main programs only.)

3b. An "ENTRY" pseudo-instruction and a page title card. (Procedures only.)

4. Pseudo-operations defining non-standard operation codes used in compiled MILITRAN programs.

5. Common storage allocation for use by MILITRAN Object-Time Library routines.

6. Common storage allocation for variables in the source program, if required.

7. A one-word constant which provides a link between local and common OBJECT tables.

8. Tables defining OBJECTS in local storage, if required.

9. System constant locations, if required.

10. Local storage allocation or common storage values for TIME, ATTACKER, TARGET, and INDEX if required.

11. Local storage allocation and common storage values for all variables in the program as required.

12. FORMATS specified in the source program.

13. FORMATS generated by the processor for listing of symbolic dimensions. (Items 13 thru 21 apply to main programs only.)

14. A FORMAT for reading of symbolic dimensions.

15. Symbol-defining pseudo-operations
    for use in reading and writing
    symbolic dimensions.

16. Instructions which initialize the
    floating-point trap routine and
    common storage tables.

17. Instructions which read and list
    symbolic dimensions.

18. Calculation of dimension values
    which are arithmetic expressions.

19. Storage of dimension values in
    reference tables for arrays, vectors,
    lists, and objects.

20. Adjustment of tables for classes
    whose members have symbolic dimensions.

21. Allocation of storage for arrays,
    vectors, and lists whose storage
    requirements depend upon symbolic
    dimension values.

22. Processing specified by executable
    statements in the source program.

23. Exit sequence to monitor for main
    programs, to calling program for
    procedures.

24. Instructions establishing communi-
    cation between a procedure and its
    calling sequence. (Procedures only.)

25. Allocation of temporary storage
    locations.

26. "END" card.

Diagnostics issued during Pass IV processing are
stored on Scratch Tape 1 until generation of the FAP pro-
gram is complete. They are then retrieved and expanded
onto the listing tape.

All scratch tapes are rewound at the end of
Pass IV and control is returned to Pass I for processing
of the next source program.

APPENDIX 4

MILITRAN PROCESSOR OPERATING SUMMARY

Tapes

MILITRAN Compiler Tape on A4, file protected.

Scratch tapes on A5 and B5.

Printer Comments

BEGIN MILITRAN COMPILATION

NO ERRORS IN ABOVE COMPILATION

xx ERRORS IN ABOVE COMPILATION

END OF TAPE A3

BAD A4

Error Stops

Location 00024: Redundancy. Can not be ignored. Kill job.

Location 00025: End of tape. Can not be ignored. Kill job.

Unexpected Stops

1. Switch to manual.

2. Place 7xx xxx 000021 in keys, where
    xxxxx is location counter.

3. Press "ENTER INSTRUCTION"

4. Switch to automatic

5. Press "START"

6. Clear keys after 10 seconds.

FORTRAN Monitor Operations

Before attempting standard monitor procedures such as
dumps or skipping to next job, always press RESET.

INDEX

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*	2a. REPORT SECURITY CLASSIFICATION
Systems Research Group, Inc. 1501 Franklin Ave. Mineola, Long Island, N. Y.	Unclassifed
	2b. GROUP

3. REPORT TITLE

MILITRAN OPERATIONS MANUAL FOR IBM 7090-94

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Technical report

5. AUTHOR(S) *(Last name, first name, initial)*

Systems Research Group, Inc.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
June 1964	69	

8a. CONTRACT OR GRANT NO. Nonr 2936(00)	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO. Navy NR 276-001	
c. AF Proj. 2801,	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* USAF Technical Documentary
d. Task 280101	Report No. ESD-TDR-64-389

10. AVAILABILITY/LIMITATION NOTICES
Qualified requesters may obtain copies of this report from the Defense Documentation Center (DDC)

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Office of Naval Research, Wash. D.C. & Electronic Systems Division, Air Force Systems Command, Bedford, Mass.

13. ABSTRACT

MILITRAN is an algorithmic computer language specifically oriented to the problems encountered in simulation programming. In addition to providing overall flexibility in expressing complex procedures, the language contains features which greatly simplify the maintainence of status lists, handling of numeric and non-numeric data, and sequencing of events in simulated time.

This report describes the features and operating procedures of the 7090-94 MILITRAN Processor.

DD FORM 1473
JAN 64

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Militran						
Language						
Simulation						
Computers						
Programming Languages						
Data Processing Systems						
Information Retrieval						
Instruction Manuals						
Compiler						
Systems Analysis						
War Gaming						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization (corporate author) issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers (either by the originator or by the sponsor), also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those

imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through

_____ ."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through

_____ ."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through

_____ ."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring (paying for) the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as (TS), (S), (C), or (U).

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.